Be a Collaborator and a Competitor in Crowdsourcing System

Iheb Ben Amor
Université Paris Sorbone Cité
Paris Descartes
France
Email: iheb.ben-amor
@etu.parisdescartes.fr

Mourad Ouziri and Soror Sahri
Université Paris Sorbone Cité
Paris Descartes
France
Email: firstname.lastname
@parisdescartes.fr

Naouel Karam
Freie Universität Berlin
Germany
Email: naouel.karam
@fu-berlin.de

***Abstract*—Crowdsourcing is emerging as a powerful paradigm to solve a wide range of tedious and complex problems in various enterprise applications. It spawns the issue of finding the unknown collaborative and competitive group of solvers. The formation of collaborative team should provide the best solution and treat that solution as a trade secret avoiding data leak between competitive teams due to reward behind the outsourcing of the issue. The formation of effective competitive teams not only requires adequate skills between members of each team, but also good team connectivity through social network and to provide the best solution and treat that solution as a trade secret avoiding data leak between teams due to reward behind the outsourcing of the issue. In this paper, we propose a data leak aware crowdsourcing system called *SocialCrowd*. We introduce a clustering algorithm that uses social relationships between crowd workers to discover all possible teams while avoiding inter-team data leakage.**

I. Introduction

A new trend of teamwork has been emerged unconstrained by local geography, available skill set, networking and deep relationships the *crowdsourcing*. It is the action of outsourcing tasks, traditionally performed by an employee or contractor, to an undefined group of people through an open call [3]. Crowdsourcing applications should be enable to seek for people crowd on demand to perform a wide range of complex and difficult tasks. Thousands human actors will provide their skills and capabilities in response to the call.

We introduce a type of crowdsourcing called *SocialCrowd* based on the efficiency of social network to outsource a task to be performed by people on demand instead of an open world as Mechanical Turk is doing. In fact, the interactions between people involved to answer a query become complex more and more and the collaboration leads to the emergence of social relations and a social network can be weaved for human-task environment.

The goal of SocialCrowd system is to let people collaborating on a joint task in the crowd environment where they may seek for other members towards social crowd relationships for achieving a business goal. Thus, many competitive teams can provide a set of answers to the call. However, as the crowd task is competitive between teams, it is important to group people in a manner there is no inter-teams leaking. Such mechanism will avoid the information leak between crowd people in different teams. Hence, the issues and challenges considered in our system, include:

(i) how to build up and discover teams answering the query towards the social relationships.

(ii) how to avoid the solution disclosure of the problem during the teams construction between competitive groups.

In fact, people on demand collaborating to a task may share sensitive information (part of the problem solution)that may be propagated or forwarded to other crowd members in the social network.

Few works dealing with crowdsourcing are provided. In [7], the Trivia Master system generates a very large Database of facts in a variety of topics, cleans it towards a game mechanism and uses it for question answering. In [14] is proposed a novel approach for integrating human capabilities in crowd process flows.

In [10]is introduced an answering queries with Crowdsourcing. The authors explain their challenges and present an initial solution to the problems of finding new data and comparing data. For this, they propose a simple SQL schema and query extensions that enable the integration of crowdsourced data and processing, they present also the design of croudDB including new crowdsourced query operators and plan generation techniques that combine crowdsourced and traditional query operators. After that, they describe methods for automatically generating effective user interfaces for crowdsourced tasks, and present the result of micro-benchmarks of performance of individual crowdsourced query operators on the ATM platform and demonstrate that CrowdDB in indeed able to answer queries that traditional DBMS cannot.

In [9] designed a wizard that may automatically configure a user's privacy settings with minimal effort from the user to aim policy preferences learning, to classify users and an automatic access rules affectation to new user. They developed a tool, an easy to use and provides very simple user interactions, it leveraging the machine learning paradigm of active learning, it interactively asks the user to assign privacy labels to specific, carefully selected, friends. As the

user provides more input, the quality of the classifier improves.

[17] models the relationship strength in on-line social networks, the authors develop an unsupervised model to estimate relationship strength from interaction activity and user similarity. Them model can represent the full spectrum of relationship strength and propose an unsupervised method to infer a continuous-valued relation-ship strength links since the relationship strength directly impact the nature and frequency of on-line interactions between a pair of users so impact the privacy policies of each user.

In [15], the authors introduced privacy protection tool that measures the amount of sensitive information leakage in a user profile and suggest self-sanitization action to regulate the amount of leakage. The protection tool determines the probability of individual sensitive attribute inference in a user profile based on her/his friendship relations and friend's attribute values.

In [4] has concluded the need of more flexible mechanisms of protection, making a user able to decide which network participants are authorized to access his/her resources and personal information to enforce privacy policies. Hence, they focused their work on relationship protection by proposing a strategy exploiting cryptographic techniques to enforce a selective dissemination of information concerning relationships across a social networks.

In [18] the authors introduce the problem of image search. Even when we have to deal with variations in lighting, texture, image quality, the search should be precise and return very few erroneous results. They wish to return one validate image before deadline, while minimizing the money. To do this, they propose a solution named CrowdSearch, in order to provide an accurate image search system for mobile devices by combining an automated Image search and then a human validation based on crowdsourcing that increases search result accuracy. The combination presents a complex set of trade-offs involving delay, accuracy and cost.

In [16] they are interested in the problems of finding all duplicate records. They propose an hybrid human machine system for entity resolution. They uses machine based techniques to discard those pair of records that look very dissimilar and only asks the world to verify the remaining pairs which need human insight. They combine the efficiency of the machine based approaches with the answer quality obtained from people. They uses heuristics approach to fight against the problem of processing time.

In [5] are interested in the problems of linking entity from text to the linked open data cloud. They linked definitions to words in texts. To do this, they propose the ZenCrowd allowing linking definitions to words in texts. They combine both algorithmic and manual linking, automate manual linking via crowdsourcing and dynamically assess human workers with a probabilistic reasoning framework.

In [6], the authors present an hybrid human machine pipeline, a novel system used for answering complex keyword queries. Using the crowd to understand the query (gain knowledge of query structure and entity relationships). They develop a new language to answer the queries.

[2] develops how to exploit the unique capabilities of the crowd to mine data from the crowd. They define a method of accessing personal databases based on association rules. The definitions of support and confidence are used as a measure for the significance of a rule per user.

The work in [8] is based on the social networks where the proposed systems searches the most suitable worker to answer the task. It is proposed a model based on the workers' profiles using information on social network platforms. The authors developed a native Facebook application named Open Turk implementing tasks assignment.

In [13] it is introduced the problem of the Interactive Crowdsourcing, they present the smartcrowd a framework for harnessing the crowd to approximate ground truth effectively and efficiently, while taking into account the innate uncertainty of human behavior. They formulate task assignment as an optimization problem, and rely on pre-indexing workers and maintain the indexes adaptively, in such a way that the task assignment process gets optimized both qualitatively, and computation time-wise. They present rigorous theoretical analyses of the optimization problem and propose optimal and approximation algorithms.

Privacy and data leaking are not at all discussed in these works.

Contributions:

We address the aforementioned challenges by proposing the SocialCrowd framework to discover competitive and collaborative composition teams answering the query through a social network while prohibit data leak between those teams.

The discovering method (DLTD) is a clustering algorithm to classify the potential crowd people from the social network that are *close* to collaborate in the same team. The system will not group them in the competitive clusters, thus, avoid inter-teams data leaking.

To handle such leak, the HDPD algorithm, a Markov chain-based algorithm, is proposed to discover the implicit social relationships between crowd people.

The rest of the paper is organized as follows: section 2 provides an overview of our *SocialCrowd* system. The propagation process is described in section 3. In section 4 is discussed the clustering based approach to discover competitive clusters in the social network answering a business call. We discuss our experiments in Section V.

II. *SocialCrowd*: An Overview

The *SocialCrowd* architecture is composed of three main components as depicted in Fig.1:

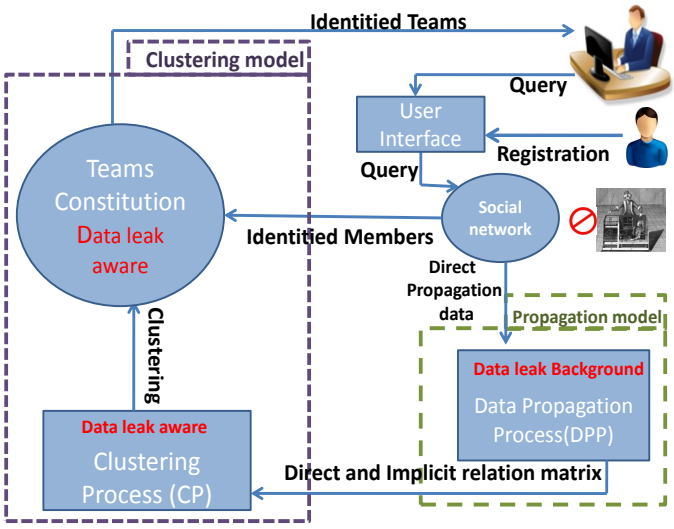

Fig. 1. *SocialCrowd* architecture

- **Data propagation component**:

  In order to compose data leak free teams, the data propagation component computes the propagation of data in the social network. The component uses the collected (inputs) and shared (outputs) information of each social member. The proposed data propagation algorithm HDPD (High Data Propagation Discovery) deals with the dynamic aspects of the network.

  It first computes the probabilities of data propagation from each member to direct friends in the social network. Then, it calculates data propagation probabilities from each member to indirect friends.It is a based Markov Chain model detailed in next section.

- **Clustering component**:

  Given the data propagation probabilities provided in the previous step, the clustering component groups crowd workers[1] into competing clusters (or teams) free of data leak. For that purpose, we define a distance to capture data leaks.

  Crowd workers with higher propagation probability are part of the same cluster. The proposed clustering algorithm DLTD (Data Leak free Team Discovery) returns all potential data leak free teams. Details about the DLTD algorithm are given in Section IV.

- **Team Constitution**:

  The module will constitute the different team based on the provided result from the clustering process (CP). It will use the user profile information provided from the social network. It will notify the users about the team discovering results.

[1]In the rest of the paper, we use the terms "member" and "crowd worker" interchangeably.

III. A Markov chain-based approach for data propagation

The aim of our approach is to build up teams from the social networks, all the while avoiding propagation of data between competitive clusters. For that, the clustering algorithm needs all the possible interactions between members of a social network.

A Social Network is set of direct relationships between members. These direct relationships allow us to compute the probability of data propagation (and consequently any data leak) between only direct friends. However, to discover competitive teams that are data leak aware, one needs to know all possible data propagations from friend-to-friend in the whole social network.

The Markov chain model is used, because

- in real world the data propagation depends only of the present state and not the past one.
- The discovering of the maximum data propagation between members is probabilistic.

The data share is an imminent information to identify competitive teams (by prohibiting leakage between discovered clusters) and collaborative teams (by ensuring that the members in the cluster interact well).

In real social network, data are shared from friend to friend and so on, so the friendship relation is a good indicator of data share in our case.

As example, the social network of Fig. 2 shows that Alice shares 90% of his data with Mickael. This rate is considered as probability of data propagations between direct friends noted by $p_{(Alice,Mickael)} = 0.9$. However, the data propagation to indirect friends (to friends-of-friends) are implicit. They depend on data propagation probabilities between direct friends. In the Fig. 2, the probability of data propagation from Alice to indirect member David is computed by the expression:

$$p_{(Alice,David)} = p_{(Alice,Mickael)} \times p_{(Mickael,David)} = 0.9 \times 0.5 = 0.45.$$

All the possible paths in the social network have to be explored to compute data propagation between two members. This is a hard computing task as real social networks are quite complex.

In this section, we present a Markov chain-based approach for handling the implicit/indirect relations between members. We present a model and an algorithm of data propagation across the entire social network.

A. A graph-based model of data sharing relationships

In this paper, we model the relationships in the social network without focusing on their nature (e.g., friendship, sharing, etc.). We only need to reason on the quantity of data propagated between members in the social network. Thus,

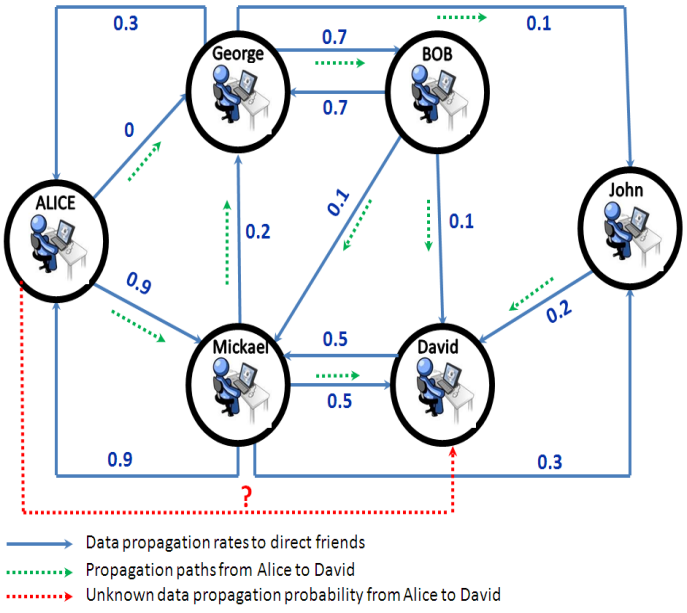

Fig. 2. A simple data propagation

hereafter we denote all kinds of relationships by the friendship relation.

We model the network as a labeled directed graph $G\langle M, A, P\rangle$ where :

- $M = \{m_i\}$: set of nodes where each node represents a social network member.
- $A = \{a_{ij} = (m_i, m_j)/(m_i, m_j) \in M\}$: set of edges where each edge represents a relationship between two members.
- $P = \{p_{ij}/\forall i, j\ p_{ij} \in [0, 1]\}$: is a set of labels where each label p_{ij} of the edge a_{ij} represents the rate/probability of data shared by the member $m_i \in M$ with his friend member $m_j \in M$.

In the given graph model, the *friend* relationship is represented using edge A labeled with the real probability of shared data P. We calculate the probability p_{ij} that the member m_i shares his data with member m_j by the simple formula:

$$p_{ij} = \frac{quantity\ of\ data\ that\ m_i\ shared\ with\ m_j}{quantity\ of\ data\ held\ by\ m_i}$$

where we consider the data held by the member m_i as only the data that m_i received from other direct members in the social network. For instance in Fig. 2, the arc $(Alice, Mickael)$ indicates that $Alice$ has a relationship with $Mickael$, and shares with him 90% of his data and the arc $(Alice, George)$ indicates that $Alice$ has a relationship with $George$ but she never shares any data with him.

In the following, we present the Markov propagation model to compute the implicit probability of data propagation between indirect friends (e.g., the propagation probability from $Alice$ to $David$ in Fig. 2).

B. Markov chain-based model for data propagation

Given an owned data of a known member, the aim of this section is to compute the propagation probabilities to all the possible members of the social network. The premise is that in social networks data is propagated from one friend to another and so on. This observation about propagation of owned data from friend to friend corresponds to the well known Markov chain [11].

Definition 1: (Markov chain)

A Markov chain is a sequence of random variables $X_1, ..., X_n$ with the Markov property; that the future state depends only on the the present state, and not on the past states. Formally,

$$P(X_{n+1} = x|X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = P(X_{n+1} = x|X_n = x_n)$$

From this formal definition, the probability that a given member gets some data depends on the probability to get it from only his direct friends (and not from indirect friends).

We represent the probability of data propagation between direct friends as a Propagation Matrix (defined as follows).

Definition 2: (Propagation matrix)

The Propagation Matrix of a social network $G\langle M, A, P\rangle$ is a matrix that gives the probability p_{ij} of propagating data between each couple of friend members (m_i, m_j), where

$$p_{ij} = \begin{cases} \frac{quantity\ of\ data\ that\ m_i\ shared\ with\ m_j}{quantity\ of\ data\ held\ by\ m_i} & \text{if } (m_i, m_j) \in A \\ 1 & \text{for } i = j \\ 0 & \text{else} \end{cases}$$

This propagation matrix has the following properties:

- $p_{ii} = 1$ means that member m_i does not lose the owned due to sharing.
- $\sum_{k\in[1,n]}(p_{ik}) \neq 1$, as data may be propagated to several members at the same time.
- $p_{ij} \neq p_{ji}$, which means that a member m_i may share data with a friend m_j in a quantity that is different from what his friend m_j may share with him.
- $(m_i, m_j) \in A \not\Rightarrow p_{ij} \neq 0$, which means that members do not necessarily share their data with their friends.
- $(m_i, m_j) \notin A \Rightarrow p_{ij} = 0$, to express that data propagation to indirect friends in the social network is unknown.

Based on the above mentioned definitions, the propagation matrix of the social network of Fig. 2 is given as follows:

$$\begin{array}{c} \\ Bob \\ Alice \\ John \\ David \\ Mickael \\ George \end{array} \begin{array}{c} \begin{array}{cccccc} Bob & Alice & John & David & Mickael & George \end{array} \\ \left(\begin{array}{cccccc} 1 & 0 & 0 & 0.1 & 0.1 & 0.7 \\ 0 & 1 & 0 & 0 & 0.9 & 0 \\ 0 & 0 & 1 & 0.2 & 0 & 0 \\ 0 & 0 & 0.2 & 1 & 0,5 & 0 \\ 0 & 0.9 & 0.3 & 0.5 & 1 & 0.2 \\ 0.7 & 0.3 & 0.1 & 0 & 0 & 1 \end{array} \right) \end{array}$$

C. A Markov chain-based algorithm of data propagation

The propagation matrix of Section III-A gives only the probability of data propagation between direct friends. But the matrix is not sufficient to compute the probability that data is propagated from one member to other indirect-friend(s) because:

1) The propagation probability between direct friends is not optimal. Following different paths in the graph of the social network, we can calculate different probabilities of propagating data between two members.

   In Fig. 2, the direct friend relationship from $Alice$ to $George$ indicates that $Alice$'s data is not propagated to George as $p_{(Alice,George)} = 0$. However, through $Mickael$, $Alice$'s data may be propagated to $George$ with probability $0.9 \times 0.2 = 0.18$.

2) Propagation to indirect-friends is not given in the propagation matrix and is set to zero. The probability of sharing data with indirect friends is zero in the propagation matrix because members share their data only with direct friends. However, data may be propagated from direct friends to indirect-friends.

   In the propagation matrix corresponding to Fig. 2, probability that $George$'s data will be propagated to indirect-friend $David$ is zero because $George$ is not a direct friend. However, $George$'s data may be propagated to $David$ through Bob with probability $0.7 \times 0.1 = 0.07$.

3) Computing propagation to indirect-friends is hard. The probability that $Alice$'s data may be propagated to $David$ (probability of dotted red arrow in Fig. 2) is hard to compute because it requires exploring all the propagation paths from $Alice$ to $David$ (dotted green arrows in Fig. 2 - although the graph of Fig. 2 is small, there are seven paths).

   All the possible paths in the social network have to be explored to compute data propagation between two members. This process is not straight-forward as real social networks are quite complex.

In this regard, we propose an algorithm to calculate the optimal probability of data propagation from a given member to all the members of the social network. This algorithm is based on an energy function we define as follows:

Definition 3: (Energy function)

The energy function p_i is the probability that data is propagated to the member m_i. Since in our model data is propagated following a Markov chain:

$$p_i = \underset{m_k \in N_{m_i}}{Max} \ (p_k \times p_{ki}) \quad (1)$$

where,
N_{m_i} is a set of direct friends of m_i, p_k is the energy function of m_k and p_{ki} is the probability of propagating data from m_k to m_i.

The maximum function is the suitable aggregation function as the aim of the approach is to estimate the maximum risk of data propagation. The aim is to compute the data probability propagation p_{ij} and p_{ji} of all the pairs of members (m_i, m_j). This requires the use of an iterative algorithm. The proposed HDPD algorithm computes the optimal energy functions $P_{o*} = (p_{o1}, p_{o2}, ..., p_{on})$, which represent probabilities of data propagation from the member m_o to members $\{m_i\}_{i \in [1,n]}$. It processes as follows:

- Initialisation:$p_{owner} = 1, \forall i \neq owner \ p_i = 0$. That is only the owner has the data.
- Iterations: At each iteration, the algorithm computes p_i for $m_i \in N_{m_i}$using the energy function.
- Stops: The algorithm stops when the maximum probability of each member is reached.

The algorithm is recursive, the complexity of the algorithm is the main call. Given, n the number of registered members, m the number of relations and $f(n)$ the main call propagation function complexity. This function is composed of a test of complexity O (n) and it makes n recursive call. Then, the complexity of this function is : $O(n * m) \times n$.

Our algorithm is presented as follows:

Algorithm 1 HIGH DATA PROPAGATION DISCOVERY (HDPD)

Require: $G\langle M, A, PM\rangle$ – labeled directed graph of the social network where PM is the propagation matrix
m_o – owner of the data
Ensure: P – all the implicit data propagation from m_o
1: $\mathcal{P} = (p_1, \ldots, p_o, \ldots, p_n)$ – Energy function at the current step.
2: $\mathcal{PS} = (ps_1, \ldots, ps_o, \ldots, ps_n)$ – Energy function at the next step.
3: *continue*: boolean value indicating if the optimal probabilities are reached.
{I}nitializations
4: $p_o = 1$ and $\forall i \neq o, p_i = 0$
5: $continue \leftarrow true$
{I}terations
6: **while** *continue* **do**
7: **for** each members $m_i \neq m_o$ **do**
8: $ps_i = Max_{m_k \in N_{m_i}}(p_k \times p_{ki}))$
9: **end for**
10: **if** $\mathcal{P} \neq \mathcal{PS}$ **then**
11: $p_i \leftarrow Max\,(ps_i, p_i)$
12: **else**
13: $continue \leftarrow false$
14: **end if**
15: **end while**
16: **return** P

Running the algorithm on the propagation matrix of previous subsection III-B results in the following graph (but only some links are shown for readability):

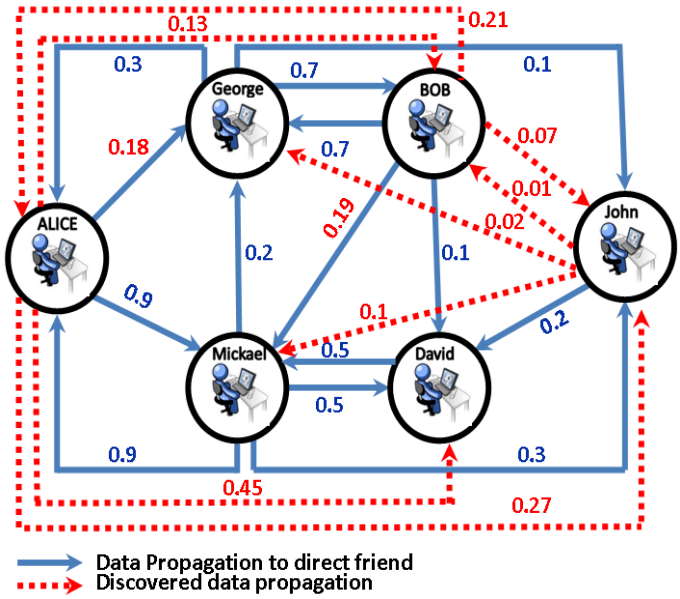

Fig. 3. Data propagation graph with implicit relationships

This graph shows that:

- For indirect friend relationships such as $(Alice, David)$ and $(Alice, John)$, the algorithm calculated data propagation probabilities that do not exist in the initial graph of Fig. 2.
- For some direct friend relationships such as $(Alice, George)$ and $(Bob, Mickael)$, the algorithm updates the data propagation probabilities from 0 and 0.1 to the optimal values 0.18 and 0.19 respectively. These optimal probabilities are calculated using the paths $(Alice, Mickael, George)$ and $(Bob, George, Alice, Mickael)$, respectively.

IV. DATA LEAK AWARE CLUSTERING

In this section, we provide details on the clustering process that groups crowd members into data leak free clusters, based on the data propagation technique defined in the previous section. In order, to give chance to every registered member in the competition, the clustering process discover all possible solutions including each member only to one cluster. In contrast to existing clustering algorithms (such as k-means [12] and our previous presented algorithm [1]) that generate a single clustering solution, our proposed approach generates all possible clustering solutions while preserving their respective data.

Before presenting the algorithm details, we give the following definitions:

Definition 4: (Cluster)

A cluster C is set of crowd members (having propagation, or no strong propagation):

$$C = \{m_i\} \text{ such that } \forall m_i, m_j \in C, p_{ij} \in [0,1], p_{ji} \in [0,1]$$

Definition 5: (Data leak free clusters)

Two clusters C_k and C_s are data leak free iff:

$$\forall m_i \in C_k, \forall m_j \in C_s, p_{ij} \leq \eta \wedge p_{ji} \leq \eta$$

In definition 5, we consider there is a risk of data leak between two clusters C_k and C_s if the propagation rate between all the members of the two clusters is less than a threshold η. The clustering algorithm uses the following definition (of distance) to generate data leak free clusters:

Definition 6: (Distance)

The data propagation between a member m_i and a cluster C_k is measured by the distance $\Omega(C_k, m_i)$, defined as follows :

$$\Omega(C_k, m_i) = \underset{m_j \in C_k}{Max}\ p_{ij}$$

It follows from definitions 5 and 6 that there is a data leak from a cluster C_k to member m_i if: $\Omega(C_k, m_i) > \eta$.

Definition 7: (Clustering solution)

A solution $S = \{C_1, ..., C_n\}_{n \geq 1}$ is set of clusters such that:

$$\forall i, j \in [1, n]\ C_i \text{ and } C_j \text{ are data leak free.}$$

Note that a solution is incomplete if it does not contain all the members, and complete otherwise.

Using definition 5, we transform the propagation matrix, computed by the HDPD algorithm, into a symmetric matrix as follows:

$$\forall i, j, p_{ij} = Max(p_{ij}, p_{ji})$$

The propagation matrix will be modified such as considering now an undirected graph and update the relationships between two users with the high value of their propagation. The maximum will be the criteria used to avoid data leaking between clusters.

As mentioned earlier, the Data Leak free Team Discovery algorithm (DLTD) will generate all the possible clustering solutions based on the defined symmetric matrix. The processes are as follows:

Initialization:

- A random member m_0 is selected, and made the member of a new cluster C_1. That is, $C_1 = \{m_0\}$
- The initial partial solution is created as $S_1 = \{C_1\}$.

Using the partial solution(s) $S_i = \{C_1, ..., C_n\}_{n \geq 1}$ (i is the number of clustered members in S_i), the algorithm clusters a new member m_i according to the following rules:

Rule 1: If m_i has data leak with only one cluster C_k in S_i (i.e., $\exists C_k \in S_i \Omega(C_k, m_i) > \eta$ and $\forall C_{jj \neq k} \in S_i, \Omega(C_j, m_i) \leq \eta$), then the member m_i is clustered into C_k. The partial solution S_i is increased to S_{i+1} such that $S_{i+1} = S_i$ where $C_k = C_k \cup \{m_i\}$.

Rule 2: If m_i has no data leak with any cluster in S_i (i.e., $\forall C_j \in S_i, \Omega(C_j, m_i) > \eta$), then the member is clustered in each cluster while creating a new solution for each instance.

That is, S_i increases to n possible partial solutions S^{α}_{i+1}: $S^{\alpha}_{i+1} = S_i \cup \{C_\alpha\}$

where $C_\alpha = C_\alpha \cup \{m_i\}$ for $\alpha \in [1, n]$.

Rule 3: If m_i has data leak with two or more clusters in S_i, i.e., for SC_i in S_i, $SC_i = \{C_1, ...C_k\}_{2 \leq k \leq n}$ such that $\forall C_j \in SC_i, \Omega(C_j, m_i) > \eta$, then there are two possible solutions (S^1_{i+1} and S^2_{i+1}):

(1)Merge the clusters $\{C_1, ...C_k\}$ and m_i into one cluster $\bigcup_{i=1,k} C_i \cup \{m_i\}$ and The partial solution S_i is increased to S^1_{i+1}

(2) Compute the subset $L \subseteq \bigcup_{C_i \in SC_i} C_i$ such that members of $L \cup \{m_i\}$ have direct or indirect data leak between them and Create a new cluster $C_g = L \cup \{m_i\}$ and remove members of L from their respective cluster finally the partial solution S_i increased to S^2_{i+1} $S^2_{i+1} = S_i \cup \{C_g\}$.

The DLTD clustering algorithm applies the above mentioned defined rules until all members are clustered in all possible solutions. The algorithm is traced as follows:

Algorithm 2 DATA LEAK FREE TEAM DISCOVERY

Input: The symmetric propagation matrix, M : set of all crowd members to be clustered.
Output: All possible solutions S_i of data free clustering. {I}nitializations
randomly select m_0 from M;
create the cluster $C_1 = m_0$;
$S_1 = C_1$ **call** FreeLeakClustering $(S_1, M - m_0)$; *Clustering function* **Function** *FreeLeakClustering*(Solution S_i, Members M);
if M = $\emptyset$ **then**
  Return S_i;
  $\backslash S_i$ is a final solution as there are no remaining members to cluster
end if
if Rule1: m_i has data leak with only one cluster C_k in S_i **then**
  $C_k = C_k \cup m_i$
  $S^j_{i+1} = S_i$
  call *FreeLeakClustering*$(S_{i+1}, M - m_i)$;
end if
if Rule2: m_i has no data leak with any cluster in S_i **then**
  for each cluster C_j in S_i **do**
    $C_j = C_j \cup m_i$
    $S^j_{i+1} = S_i$
    call *FreeLeakClustering*$(S^j_{i+1}, M - m_i)$;
  end for
end if
if Rule3: m_i has data leak with two or more cluster SC_i in S_i, $SC_i = C_1, \ldots, C_{k2 \leq k \leq n} \subseteq S_i$ **then**
  $\backslash$ First alternative solution Merge all the clusters of SC_i with m_i into the new cluster C_g. that is,
  $C_g = C_1 \cup \ldots \cup C_k \cup m_i$;
  call *FreeLeakClustering*$(S^1_{i+1}, M - m_i)$;
  $\backslash$ Second alternative solution compute the subset $L \subseteq \bigcup_{C_i} \in SC_i\ C_i$ of members having direct or indirect data leak between them 'see Rule3.2);
  Create a new cluster $C_g = L \bigcup m_i$;
  for m_j in L **do**
    Delete m_j from its cluster in S_i;
  end for
  $S^2_{i+1} = S_i \bigcup C_g$
  call *FreeLeakClustering*$(S^2_{i+1}, M - m_i)$;
end if

As example, Fig. 4 shows the resolution graph generated by DLTD clustering algorithm when running on the graph of Fig. 2. Internal nodes ($S^*_i, i \leq 5$) are partial solutions and leaf nodes (S^*_6) are final solutions as all members are clustered (there are 6 members to be clustered for this example).

The graph shows five solutions of clustering. In each solution, members of the social network are grouped into data leak free clusters, which can be easily checked using symmetric propagation matrix given at the beginning of this section.

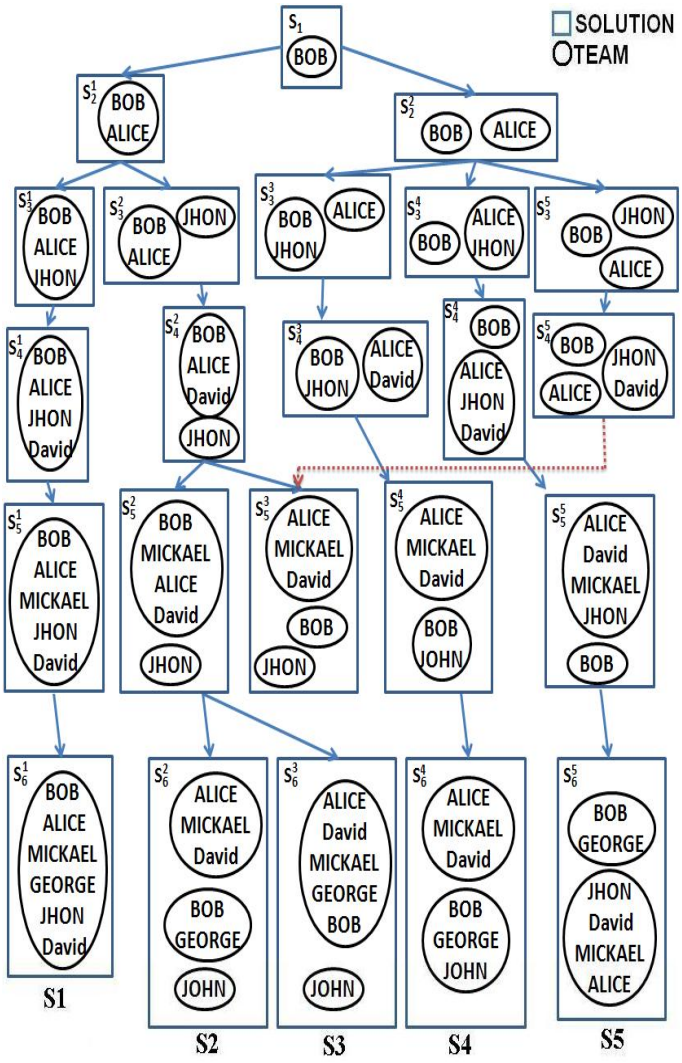

Fig. 4. Resolution graph generated by the clustering algorithm

The algorithm is recursive and consists of three main rules: Rule 1, Rule 2 and Rule 3. The complexity of the algorithm is the main call. Given, n the number of members to be classified, and $f(n)$ the main call clustering function complexity.

Then, $f(n) = Max(O(Rule1(n)), O(Rules2(n)), O(Rule3(n))$.

1) Complexity of rule 1 : This rule is composed of a test of complexity $O(n)$, and a recursive call.

   Then, the complexity of this rule is : $O(n)+O(f(n-1)) = n+(n-1)+O(f(n-2)) = n+(n-1)+\ldots+O(1) = O(n^2)$.

2) Complexity of rule 2 : This rule makes k recursive call where k is the number of clusters in a solution.

   Then, the complexity of this rule is : $O(n)+k\times O(f(n-1)) = n+k\times(k-1)\times\ldots\times 1 = O(n+k!)$.

   If we consider the theoretical worst case $k = n$, then complexity of rule 2 is $O(n!)$.

3) Complexity of rule 3 : This rule is composed of a test of complexity O (n), and two recursive calls.

   Then, the complexity of this rule is: $O(n)+2\times O(f(n-1)) = n+2(n-1)+O(f(n-2)) = n+2(n-1)+\ldots+2 = O(n^2)$.

Hence, the overall complexity of the algorithm is $O(n!)$. This is a theoretical complexity in the worst case. In practice, the number of clusters k are usually much smaller than n.

V. Experiments

The experiment environment consists of a Mac OS X 10.5.8 machine with a 2.13 GHz Intel Core 2 Duo processor, and 2 GB of ram.

Coding is JAVA-based, and we have followed Metcalfe's law [2] for creating the social network.

Metcalfe's law characterizes many of the network effects of communication technologies and networks such as the Internet, social networking, and the World Wide Web. Former Chairman of the U.S. Federal Communications Commission, Reed Hundt, said that this law gives the most understanding to the workings of the Internet.

Metcalfe's Law is related to the fact that the number of unique connections in a network of a number of nodes (n) can be expressed mathematically as the triangular number n(n - 1)/2, which is proportional to n2 asymptotically.

The law is abundant and existent due to the ability of Internet users to link together. Websites and blogs such as Twitter, Facebook, and Myspace are the center of this law taking effect.

We conduct experiments on effectiveness of the proposed approach in terms of data leak comparing to K-means. K-means clustering is a well known partitioning method. In this objects are classified as belonging to one of K-groups.

The results of partitioning method are a set of K clusters, each object of data set belonging to one cluster. In each cluster there may be a centroid or a cluster representative.

A. k-means and DLTD clustering based on Ω

We present the evaluation of the proposed DLTD algorithm and compare it to the k-means clustering algorithm, based on the Ω distance.

The data leakage is calculated based on the results returned by the two algorithms. We explored each cluster in order to identify the data propagation upper than η between two

[2]Bob Metcalfe, personal communication, June 2007.

members in different clusters. To do this, we developed a script computing the leakage after the classification step.

The results are depicted in Fig. 5 (where the leakage threshold (η) is set to 40%). The figure shows that with the DLTD algorithm, the clusters are free data leak, even if, the size of the social network increases. These results are sound because the algorithm groups members leading to the data leak in the same group. In contrast, the k-means algorithm results in a higher percentage of data leaks with increasing social network size. This result is valid with varying number of clusters (k).

For instance, with 20 clusters we can see that data leak increases from 2.9% to 4.4% when the size of the social network grows from 500 to 5000 members. These results confirm that the k-means algorithm is not a good solution for handling data leaks in social networks.

In fact, even if there is high data propagation between two network members, the k-means algorithm may cluster them into different clusters due to their respective closeness in terms of the used distance.

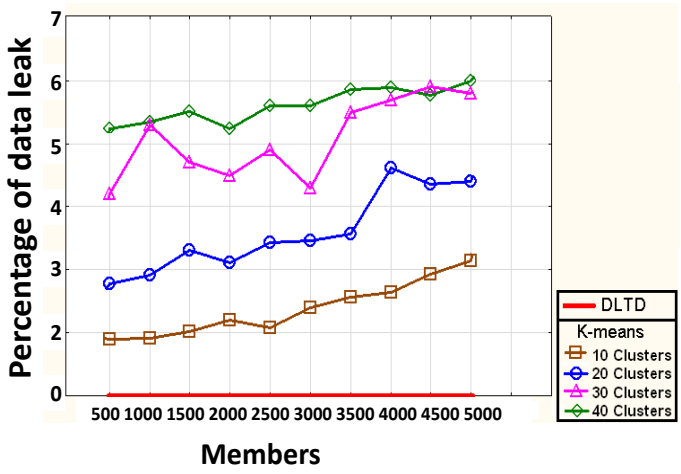

Fig. 5. Leakage comparison

B. Execution time comparison

We analysed the execution time of both DLTD en K-means algorithm using Ω distance. The analysis uses between 1000 and 5000 users. We computed the execution time of around 12497500 relationships in social network. We note that, the DLTD algorithm has no data leakage, however, its computational time is upper than the K-means.

In the figure 6, the execution time is between 153 and 1653 seconds for the k-means algorithm and it's between 172 and 2631 for the DLTD. Then, the K-means algorithm is better than the DLTD in computational time.

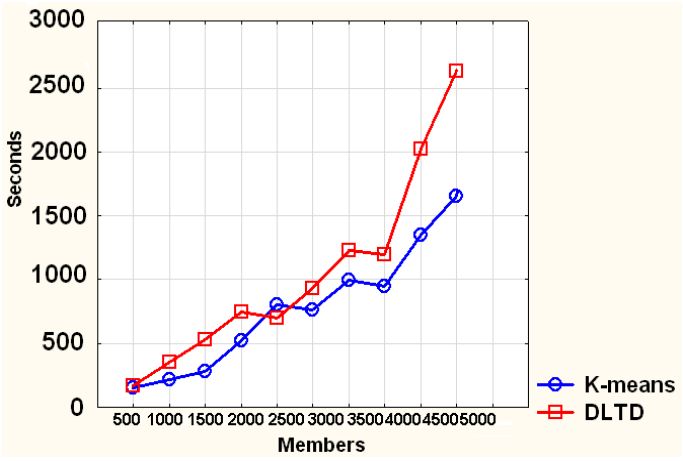

Fig. 6. DLTD and K-means execution time

The execution time of the DLTD algorithm is important than the K-means, because, the DLTD discover all possible clustering solutions, or the K-means discover only one.

Furthermore, the k-means results represent an important data leakage (data propagation upper than η) between the different clusters while using Ω distance, because, the classification mechanism group members in the nearest cluster.

This means that, k-means classify the member to the cluster with which he has the highest data propagation, even if, he has other propagations upper than the threshold with other clusters.

The execution time of the k-means algorithm is caused by the convergence process. The k-means repeating the same classification phase until convergence. The number of repetition is variable, in our case, it depends on the number of members and their propagation to classify data.

The convergence of k-means is reached when comparing the last two iterations shows that members no longer move between teams, then, the calculation of the k-means reaches its stability and no iteration is required. In addition, the convergence to a local minimum can produce a bad result.

The convergence criterion of DLTD algorithm is reached when there is no crowd member to classify.

C. Solution generation scalability

In this experiment, we shows that the DLTD algorithm returns different solutions generation depending on the number of registered member and the data propagation values and not depending on social network size.

The figure displays that more the of registered member grows more the number of generated solutions increases except from 1500 to 2000 members, because, members that has a data propagation upper than the threshold with two or

more clusters generates merging those clusters.

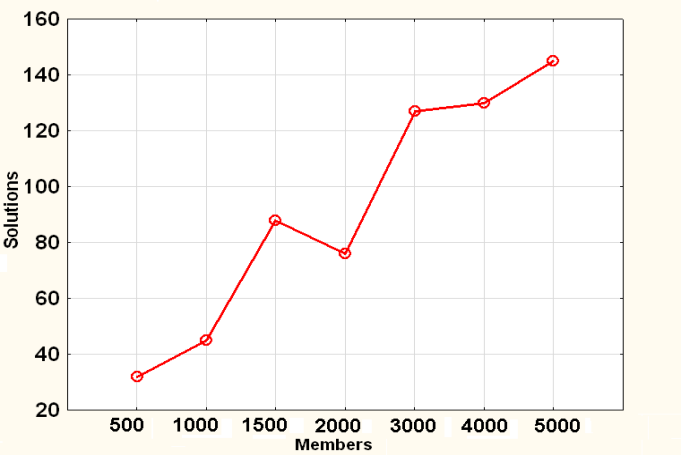

Fig. 7. Solution generation scalability

Hence, there will be less solutions. Using the same social network architecture and evolving the number of registered member, the fig 7 shows that, with 500 registered member, the DLTD algorithm return around 33 possible solutions, and for 3000 members the algorithm generate around 125 solutions.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a data leak aware crowdsourcing system providing a collaborative environment between crowd-experts organized in various teams that compete against each other to achieve certain tasks. The system prevents data leaks between teams to protect each team competitive advantage.

For that, we designed a clustering algorithm that discovers all possible teams of crowd workers while minimizing data leakage. Where the leakage is expressed by propagation of data between crowd workers via social relationships.

However, the theoretical complexity of the clustering algorithm is hard. To face with this complexity, we plan to improve algorithm by adding heuristics.

REFERENCES

[1] Iheb Ben Amor, Salima Benbernou, Mourad Ouziri, Mohamed Nadif, and Athman Bouguettaya. Data leak aware crowdsourcing in social network. In *WISE Workshops*, pages 226–236, 2012.

[2] Yael Amsterdamer, Yael Grossman, Tova Milo, and Pierre Senellart. Crowd mining. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 241–252, New York, NY, USA, 2013. ACM.

[3] Daren C. Brabham. Crowdsourcing as a model for problem solving: An introduction and cases. *Convergence: The International Journal of Research into New Media Technologies*, 14(1):75–90, 2008.

[4] Barbara Carminati, Elena Ferrari, and Andrea Perego. Enforcing access control in web-based social networks. *ACM Trans. Inf. Syst. Secur.*, 13(1), 2009.

[5] Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *WWW*, pages 469–478, 2012.

[6] Gianluca Demartini, Beth Trushkowsky, Tim Kraska, and Michael J. Franklin. Crowdq: Crowdsourced query understanding. In *CIDR*, 2013.

[7] Daniel Deutch, Ohad Greenshpan, Boris Kostenko, and Tova Milo. Using markov chain monte carlo to play trivia. In *ICDE*, pages 1308–1311, 2011.

[8] Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux. Pick-a-crowd: tell me what you like, and i'll tell you what to do. In *WWW*, pages 367–374, 2013.

[9] Lujun Fang, Heedo Kim, Kristen LeFevre, and Aaron Tami. A privacy recommendation wizard for users of social networking sites. In *ACM Conference on Computer and Communications Security*, pages 630–632, 2010.

[10] Michael J. Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. Crowddb: answering queries with crowdsourcing. In Timos K. Sellis, Renée J. Miller, Anastasios Kementsietsidis, and Yannis Velegrakis, editors, *SIGMOD Conference*, pages 61–72. ACM, 2011.

[11] Olle Hggstrm. Finite markov chains and algorithmic applications. In *in London Mathematical Society Student Texts*. Cambridge University Press, 2000.

[12] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14. California, USA, 1967.

[13] Senjuti Basu Roy, Ioanna Lykourentzou, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. Optimization in knowledge-intensive crowdsourcing. *CoRR*, abs/1401.1302, 2014.

[14] Florian Skopik, Daniel Schall, Harald Psaier, Martin Treiber, and Schahram Dustdar. Towards social crowd environments using service-oriented architectures. *it - Information Technology*, 53(3):108–116, 2011.

[15] Nilothpal Talukder, Mourad Ouzzani, Ahmed K. Elmagarmid, Hazem Elmeleegy, and Mohamed Yakout. Privometer: Privacy protection in social networks. In *ICDE Workshops*, pages 266–269, 2010.

[16] Jiannan Wang, Tim Kraska, Michael J. Franklin, and Jianhua Feng. Crowder: Crowdsourcing entity resolution. *Proc. VLDB Endow.*, 5(11):1483–1494, July 2012.

[17] Rongjing Xiang, Jennifer Neville, and Monica Rogati. Modeling relationship strength in online social networks. In *WWW*, pages 981–990, 2010.

[18] Tingxin Yan, Vikas Kumar, and Deepak Ganesan. Crowdsearch: Exploiting crowds for accurate real-time image search on mobile phones. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 77–90, New York, NY, USA, 2010. ACM.